UNITED STATES PATENT AND TRADEMARK OFFICE

[illegible]

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/712,463	11/12/2003	Judith Schwabe	P-4181CIP	9131

24209 7590 05/21/2007
GUNNISON MCKAY & HODGSON, LLP
1900 GARDEN ROAD
SUITE 220
MONTEREY, CA 93940

EXAMINER	
VU, TUAN A	
ART UNIT	**PAPER NUMBER**
2193	
MAIL DATE	**DELIVERY MODE**
05/21/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)
	10/712,463	SCHWABE ET AL.
	Examiner	Art Unit
	Tuan A. Vu	2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on 08 March 2007.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) 1-78 is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) 1-78 is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date 1/16/07.	6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 3/8/2007.

As indicated in Applicant's response, no claims have been amended. Claims 1-78 are pending in the office action.

Claim Objections

2. Claims 1, 20, 39, 58, 77, 78 are objected to because of the following informalities: the phrase 'optimizing said first instruction to a second instruction' appears to be improper use of the verb optimizing. An instruction can be optimized in that it is affected so it becomes more efficient in size or in execution resource. A conversion takes one instruction in one initial form then generates another form from it. Optimizing cannot be described in way that a first instruction is optimized **to a second instruction**. The improper language will be treated as though the first type instruction is analyzed and configured based on or relative to a second instruction type requirement in order to improve execution time.

Appropriate correction is required.

Double Patenting

3. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory obviousness-type double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent either is shown to be commonly owned with this application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement.

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

4. Claims 18, 37, 56, 75 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 12, 25, 38, 51 of U.S. Patent No. 7,107,581 (hereinafter '581). Following are examples as to how the claims are conflicting.

As per instant claim 18, '581 claim 12 also recites optimizing first instruction to a second instruction based on relationship between base of operand, the base of the second instruction smaller than that of the first instruction, converting instructions in a chain such that operands with potential overflows potential beyond the base of the second base, i.e. matching by changing type of operand of the second instruction to match with that of the first instruction base, the chain bounded by the second instruction and a third instruction being the source of the operand being subjected to the modification (see '581 claim 1); determining potential overflow associated with said second instruction and generating an output stack ('581 claim 11); indicating said 2nd instruction has potential overflow if it does not equal said first type of operand, if it does not remove such overflow, if it creates said overflow (see '581 claim 12, 1st para); indicating said 2nd instruction has potential overflow if it does not equal said first type of operand, if it does not remove such overflow, if it does not create said overflow (see '581 2nd para, claim 12: ...*if overflow is not possible*...), if said 2nd instruction propagates said overflow, and if at least one operand is one input stack has potential overflow (*).

Although '581 claim 12 does not recite 'indicating said second instruction overflow if said second type does not equal said first type ... indicating said second ... if said second instruction does not create potential overflow ... if ... propagates potential overflow ... at least one input stack has potential overflow' (as recited in instant claim 17), claim 18 and claim 17, together, can be addressed together with the subject matter of '581 claim 12. That is, the matter recited in instant claim 17 amounts to a slight variation of --yet equivalent to-- the subject matter of instant claim 18; hence would be treated as subsumed under claim 18 and therefore obvious when treating it with the '581 claim 12, as set forth above (see *).

'581 claim 12 does not recite '*one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction*'; but in view of '581 claim 12 reciting of 'determining potential overflow associated with said second instruction and generating an <u>output stack</u> based on execution of said 2nd instruction (see '581 claim 11) ... and indicating said 2nd instruction... has potential overflow, and if at least one operand ... one <u>input stack</u> has potential overflow (see '581 claim 12), the above limitation is considered analogously disclosed because output stack with operand being modified entails input stack operands potential overflow problem, according to the above from '581 to accomplish the above conversion operating on the operands associated with an input stack.

Nor does '581 recite '*changing the type of instructions ... to equal said 2nd type if said operand type is less than said second type*'; but '581 claim 1 recites that said second base (of a second processor) smaller than that of said first base so that the bounding of instructions between said 2nd instruction and a 3rd instruction -being a source of a potential overflow- is such that this chain involves converting to a wide base if the target operand (of the second instruction) carries a potential overflow that is beyond that of the second instruction. Hence, the above limitation is considered analogously disclosed by virtue of equivalent teaching albeit the apparent different wording in the language of the claims.

Instant claims 37, 56, and 75 recite the same limitations as claim 18 and '581 claims 25, 38, 51 recite the same subject matter of '581 claim 12; hence instant claims 37, 56, and 75 are also conflicting with the subject matter claimed in '581 claims 12, 38, 51 by virtue of the rationale as set forth above.

5. Claims 16, 35, 54, 73 are rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 53, 54 of U.S. Patent No. 7,107,581

Further, **instant claims 16, 35, 54, 73** conflict with the subject matter of '581 claims 53, 54 by virtue of the input stack analysis above, i.e. output stack generating based on potential overflow of operand of 2nd instruction reads on input stack associated with 1st instruction for which a 2nd instruction of operand type belonging to lower base is to matched and converted to prevent overflow.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-78 are rejected under 35 U.S.C. 103(a) as being unpatentable over Yellin et al., USPN: 5740441 (hereinafter Yellin), and further in view of Wilkinson et al., USPN: 6,308,317 (hereinafter Wilkinson).

As per claim 1, Yellin discloses a method for arithmetic expression (e.g. col. 4, lines 42-51; *integer add* - col. 9, lines 42-52; col. 17, TABLE 1: *isub, idiv, imul, iadd*) optimization, comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction (e.g. Fig. 4B-D; col. 6, lines 6-38);

optimizing said first instruction to a second instruction configured to operate on at least one operand of a second type, said second type smaller than said first type (e.g. overflow – Fig. 4C, 4D; *updated...modified ... by the current instructions* - col. 10, line 58 to col. 11, line 6; and improving execution time efficiency – col. 5, line 65 to col. 6, line 4 -- reads on optimizing), said optimizing based at least in part on the relative size of said first type and said second type (Note: overflow analysis for each successor instruction reads on size of destination place in stack for a successor operand being smaller to take required size of upper instruction in the data flow – see Fig. 4A-B); and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of instructions (e.g. Fig. 4, 5 – Note: any process that iterates one instruction at a time until all are processed reads on chain of instructions bounded by a start and a current element being processed), said chain bounded by said second instruction and a third instruction that is the source of said at least one operand (see col. 19-21: Do until there are no instructions whose changed bit is set ... Do for each successor instruction ...Merge the Virtual stack ... Verification Success – Note: algorithm to merge – see col. 11-12 --).

But Yellin does not explicitly that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. But Yellin discloses adding data to a stack of known type (see col. 1, lines 60-67; col. 20, lines 24-35; col. 21, lines 12-37; step 394, Fig. 4B) hence has suggested modifying the runtime instruction as verified by the stack to accommodate data being ported from a higher platform to a lower platform (see col. 5, lines 14-64). Analogous to Yellin's approach to modifying stack runtime Java bytecodes in order to accommodate an executing platform receiving/using data coming from a platform with higher architecture base, Wilkinson discloses modifying stack operands from a larger base platform to operands of lesser size that would fit the target platform (see Fig. 10-11; col. 11, lines 4-48). Based on the common endeavor by Yellin or Wilkinson to alleviate extraneous security resources of said receiving platform, when loading operands for each instruction in Java method prior to runtime, it would have been obvious for one skill in the art at the time the invention was made to provide the operand type replacement as approached by Wilkinson so to support Yellin's loading process. One would be motivated to do so because providing operands of smaller size and familiar to the (smaller target platform) runtime as replacement for those corresponding operands of a higher size base as purported by Wilkinson and via Yellin's attempt to obviate overflow would alleviate

Yellin's application resources for dealing with exception due to overflow, obviate the resources of the JVM interpretor role as desired by Yellin (see by Yellin: col. 15, lines 12-30), and according to Wilkinson, the stack space of the smaller platform (e.g. Fig. 21-24) might be loaded with data compliant to the platform thus expediting code execution via prechecking and reshuffling of bytecode, operands (see Fig. 9-11), enhancing the runtime with a safer method supporting a device whose resources are not equipped for checking large data being provided (see Wilkinson: BACKGROUND, col 3).

As per claims 2-3see Yellin as said first instruction is arithmetic (col. 4, lines 42-51; *integer add* - col. 9, lines 42-52; col. 17, TABLE 1: *isub, idiv, imul, iadd*); a non-arithmetic, type-sensitive instruction (see col. 16-18: Table 1).

As per claims 4-5, see Yellin (in view of Wilkinson) for repeating said validating, said optimizing and said matching for instructions that comprise a program (see Figs 4-5) and linking each instruction to input instructions in all control paths (step 366 – Fig. 4A).

As per claim 6, see Yellin (col. 5, lines 14-64) wherein said first instruction is defined for a first processor having a first base; and said second instruction is defined for a second processor having a second base.

As per claims 7-8, Yellin wherein said first processor comprises a Java Virtual Machine (see Fig. 1-2); and in view of the obviousness rationale of claim 1 see Wilkinson (e.g. Fig. 1-2) wherein said second processor comprises a Java Card Virtual Machine with resource-restraint platform, i.e. operable with lower operand type; according to which rationale, Yellin combined with Wilkinson(see Wilkinson Fig. 11) discloses wherein said first processor comprises a 32-bit processor; and said second processor comprises a resource-constrained 16-bit processor.

As per claims 9-10, Yellin discloses said at least one input stack comprises a plurality of input stacks, said plurality of input stacks further comprising a first input stack and a second input stack; and said validating further comprises comparing operand types of corresponding entries in said first input stack and said second input stack (refer to claim 1 in light of Figs. 4);, wherein said comparing further comprises indicating an error if the types of at least said first at least one stack entry and said second at least one stack entry are not equivalent (Fig. 4D, 4E, 4F).

As per claim 11, Yellin discloses wherein said optimizing further comprises: setting said second type to a smallest type; setting said second type to the type of an operand in said at least one input stack if said smallest type is less than said type of said operand (see rationale of obviousness using teaching of Wilkinson); and

setting said second type to a type that is larger than said smallest type if said smallest type is greater than said type of said operand, if said operand has potential overflow, if said second instruction is sensitive to overflow and if said second type is less than said first type (see rationale of claim 1; according to which any type if predicted to overflow beyond a smaller size will be replaced, i.e. overflow analysis as by Yellin inherently teaching setting a smallest type to a larger type when said smallest type is itself greater than type of any operand potentially risking being overflowed).

As per claim 12, Yellin (in view of Wilkinson) discloses wherein said smallest type is the smallest type supported by a target processor (see rationale of claims 7-8).

As per claims 13-14, Yellin (in view of Wilkinson) discloses wherein said smallest type is the smallest type determined during a previous pass of said optimizing (Yellin: Figs 4-5);

wherein said third instruction is not a source instruction (any instruction being loaded does not have to be a source for a operand that is predicted to overflow); and

said changing further comprises: recursively examining input instructions until said third instruction is obtained; and setting the type of said third instruction to equal said second type (Note: in view of the recursive process to verify by Yellin and changing to a smaller operand size of the target platform by Wilkinson, any replacement to match the platform operand size being included in the verification process – see Yellin: Fig. 4-5 – will read no setting a third instruction to be equal to a second instruction previously replaced in the chain)

As per claim 15, Yellin (in view of Wilkinson) discloses (see Verification algorithm: col. 19-21) wherein said third instruction comprises a source instruction (Note: within a chain of instructions source instruction is the identified instruction starting from which replacing a original first operand with target platform operand of a lesser size begins); and said

changing further comprises: recursively examining input instructions until said third instruction is obtained; setting the type of said third instruction to equal said second type; and repeating said changing for each input instruction of said third instruction (see rationale of claim 14).

As per claim 16, Yellin discloses comprising recording, said recording comprising: determining potential overflow associated with said second instruction (see col. 9, lines 30-60); but Yellin does not disclose generating an output stack based at least in part on execution of said second instruction. But Wilkinson discloses replacement of operands for a stack (see Fig. 11, 16, 18) to support Yellin's verification endeavor as set forth in claim 1; hence the stack being modified to include (as an output stack) adjusted operands as per the combination set forth in claim 1 would also be an obvious limitation to enable Yellin to provide a smaller platform to make efficient use of its runtime environment, utilizing Wilkinson's approach for the reasons as set forth above in claim 1.

As per claim 17, Yellin discloses (combined with Wilkinson) wherein said determining further comprises: indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if said second instruction creates potential overflow; and indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, if said second instruction does not create potential overflow, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow (see Yellin: col. 9, lines 30 to col. 10, line 34 – Note: determining a need for change in a chain of instructions wherein operands of smaller size would be loaded in stack to *replace* – as set forth in claim 1 -- larger size operands deemed not suitable for the target platform reads on if said second instruction *does not remove* overflow or *might not necessarily create overflow*, and the automated process by which a chain in which all instructions are recursively verified reads on propagating the replacement due to potential overflow).

As per claim 18, Yellin discloses (combined with Wilkinson)wherein said determining further comprises: indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, and if overflow is possible based at least in part on the type of said second instruction and the relationship between said first type and said second type; and indicating said second instruction has potential overflow if said second type does not equal said first type, if said second instruction does not remove potential overflow, if overflow is not possible based at least in part on the type of said second instruction and the relationship between said first type and said second type, if said second instruction propagates potential overflow, and if at least one operand in said at least one input stack has potential overflow (see Yellin in view of Wilkinson – as set forth in claim 1 -- according to the rationale of claim 17).

As per claim 19, Yellin alone does not disclose creating a new output stack based at least in part on one of said at least one input stack; updating said new output stack based at least in part on operation of said second instruction; and indicating another instruction conversion pass is required if said new stack does not equal a previous output stack. But in view of the automated and iterative verification process by Yellin combined with the operands replacement by Wilkinson as set forth in claim 1, the constant updating of stack per iteration pass in regard to what is deemed potentially conflicting in regard to stack overflow (see Yellin: Figs 4-5), it would have been obvious for one skill in the art to make many passes to update the runtime stack according to Yellin's approach using marking of unmatched operand allocation, so that combining with Wilkinson's replacement of larger size operands with more compliant size operands, Yellin's verification would be ready for execution with a stack that is repeatedly updated, and this would further alleviate runtime resources of small platforms using smaller operand type as set forth in claim 1.

As per claim 20, Yellin discloses a method for arithmetic expression optimization, comprising: step for validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction; step for optimizing said first instruction to a second instruction configured to operate on at least one operand of a second type, said second type smaller than said first type, said optimizing based at least in part on the relative size of said first type and said second type; and step for matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching including a chain being bounded; said chain bounded by said second instruction and a third instruction that is the source of said at least one operand;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly that the above matching within a bounded chain includes changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type. This limitation has been addressed in claim 1 using Wilkinson.

As per claims 21-38, refer to rejections set forth to claims 2-19 respectively.

As per claim 39, Yellin discloses a program storage device readable by a machine, embodying a program of instructions executable by the machine to perform a method for arithmetic expression optimization, the method comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

optimizing said first instruction to a second instruction configured to operate on at least one operand of a second type, said second type smaller than said first type, said optimizing based at least in part on the relative size of said first type and said second type; and matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of instructions being bounded, said chain bounded by said second instruction and a third instruction that is the source of said at least one operand;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly disclose that said matching comprises changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type; but this limitation has been addressed in claim 1.

As per claims 40-57, refer to rejections set forth to claims 2-19 respectively.

As per claims 58-76, these are the apparatus version of claims 1-19; hence are rejected with the corresponding rejections as set forth therein, respectively.

As per claim 77, Yellin discloses a method of using an application software program including arithmetic expression optimization of at least one instruction targeted to a processor, the method comprising receiving the software program on said processor, said software program optimized according to a method comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

optimizing said first instruction to a second instruction configured to operate on at least one operand of a second type, said second type smaller than said first type, said optimizing based at least in part on the relative size of said first type and said second type; and

matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of instructions being bounded, said chain bounded by said second instruction and a third instruction that is the source of said at least one operand; and executing said at least one instruction on said processor;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly disclose that said matching comprises changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type; but this limitation has been addressed in claim 1.

As per claim 78, Yellin discloses a controller configured to execute a virtual machine, the virtual machine capable of executing a software application comprising a plurality of previously optimized instructions, the instructions optimized by a method comprising:

validating at least one input stack associated with a first instruction configured to operate on at least one operand of a first type, each of said at least one input stack associated with an input instruction of said first instruction, each input stack representing the state of an operand stack associated with an input instruction upon execution of said input instruction;

optimizing said first instruction to a second instruction configured to operate on at least one operand of a second type, said second type smaller than said first type, said optimizing based at least in part on the relative size of said first type and said second type; and matching said second type with an operand type of at least one operand in said at least one input stack associated with said second instruction, said matching comprising a chain of instructions being bounded, said chain bounded by said second instruction and a third instruction that is the source of said at least one operand;

all of which limitations having been addressed in claim 1.

But Yellin does not explicitly disclose that said matching comprises changing the type of instructions in a chain of instructions to equal said second type if said operand type is less than said second type; but this limitation has been addressed in claim 1.

Nor does Yellin disclose that the controller is a smart card having a microcontroller embedded therein; but in view of the rationale as set forth in claim including Wilkinson's Java card machine, this smart card controller limitation would also have been obvious because of the benefits as set forth in that rationale.

Response to Arguments

8. Applicant's arguments filed 3/8/07 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

Double-Patenting Rejection:

(A) The Applicants have submitted that '581 claim 1 does not suggest validating, optimizing, and matching as recited in the instant claim 1 and that '581 claim 12 suggests instant claim 17 (Appl. Rmrks pg. 31, top half). The converting an instruction in a first base to a second base smaller than the first base by '581 claim 1 teaches optimizing; and the determining of type as set forth in '581 claim 12 entails inherent matching in order to validate the propriety of size to prevent potential overflow. A closer look at instant claim 17 reveals that there is no major semantic difference between instant claim 17 and claim 18; e.g. claim 18 merely an obvious rephrasing of claim 17. Accordingly, the rejection has pointed out that although '581 claim 12 does not recite the language of instant claim 17, the subject matter of claim 17 is treated as equivalent subject matter of instant claim 18, or a obvious version thereof, despite a slight variation in wording; that is, the treating of instant claim 18 (and claim 17) can be addressed together with the subject matter of '581 claim 12. Therefore, '581 claim 12 does recite or render obvious the *indicating* steps of instant claim 18 (encompassing those of claim 17). The argument is not persuasive.

(B) The Applicants have submitted that '581 claim 53 does not suggest validating, and 'elimination of the operator characteristics' as recited in the instant claim 1 (Appl. Rmrks pg. 33, bottom half). It is noted that there is no recital of 'elimination of operator characteristics' and the subject recited in claim 1 amounts to matching operand of first type with that of a second type, and in a process of configuring a operand in a first type to a second operand type, observing the respective bounds prior to converting or optimizing this first operand to make it smaller based on the above matching with the second type. '581 claim 1 and claim 53 recite a context to effectuate a equivalent scenario in which converting operand from one base to another smaller base consists of generating an optimized operand for the second base, respecting in the process an wide size overflow associated with an operand as it is determined for bounds within a chain of instructions. The double patenting rejection is an provisional rejection of obvious type. Hence as long as the respective contexts (instant claim 16 and '581 claim 53) amount to a analogous teachings (i.e. same contextual endeavor with similar implementation approaches), a slight difference in wording between the claims would have to be considered as an obvious variation of one another for the end result of the invention would be deemed same for one of ordinary skill in the art; no rebut against anticipation being at stakes here. The argument is not convincing.

35 USC § 103(a) Rejection:

(C) Applicants have submitted that Yellin does not do any optimization but instead verifies whether data types are correct, whether an overflow occurs so to prevent execution of a interpreter; and fails to teach or suggest 'optimizing a first instruction to a second instruction' (Appl. Rmrks pg. 35, middle). First, the use of 'optimizing … first instruction to a second instruction' without implementation specifics as to how optimization can be achieved would not lend to a explicit and clear context in which a process converts or translate one instruction to another. The Claims Objections as set forth above have explained that the improper use of 'optimization' will be treated as though size/type requirement of the second instruction is based upon to analyze or reconfigure the first instruction, no translation or conversion step being explicit from the above language. Second, Yellin analyzes the instructions in the context of matching size of the respective Operand stack with the required instructions being invoked in a selected set of emulated bytecodes during a verification process; and in doing this Yellin analyzes overflow condition based on the size matching and comparing, and identifies place in the bytecode program if an stack or register size conflict occurs (e.g. Fig. 4E) while updating some jump register (see Fig. 4G) upon error being identified. It is deemed that Yellin reconfigures the bytecodes while analyzing first instruction on the basis of the size requirement as to how much in size the stack operand would be needed to alleviate runtime overflow; and this 'optimizing first instruction to a second instruction configured to operate … operand of a second type … based on relative size of …first type and … second type' limitation as a whole is deemed matched by the process of observing overflow related size of bytecodes being required for the emulation and the state of the operand allocation in stack or register. The argument in regard to the limitation of 'optimizing' is therefore not sufficient to overcome the rejection.

(D) Applicants have submitted that because claim 1 entails that the second instruction is different from the first, Yellin fails to teach any change to an instruction that operates on a different operand type (Appl. Rmrks pg. 36, top). First, in order for the second instruction to be not same as a first, step has to be taken to make this translation; and in this respect the claim appears to be broad; that is, how a conversion of instruction can be explicitly described is not evident from the 'optimizing ... to a second instruction' is not sufficient from the language of the claim. Any existence of a different instruction per se on top of the first instruction with respect to the *optimizing* step is not clear. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. Second, what is recited as 'changing the type of instructions ... to equal said second type ... less than said second type' amounts to a feature for which a combined teaching and a rationale of obviousness have been put together. In response to applicant's arguments against the references individually, one cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references with motivation to combine. See *In re Keller*, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); *In re Merck & Co.*, 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986). Applicants argue that the second reference fails to correct the deficiencies of the primary reference without pointing out how exactly the combination in the §

103 would be non-obvious. This is a mere allegation, thus not satisfying a proper prima facie type of argument against a specific ground of rejection.

As a whole, the arguments are not deemed sufficient to overcome the rejection in view of the deficiencies and/or broadness in the claim language. The claims will stand rejected as set forth in the Office Action.

Conclusion

9. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Tuan A Vu
Patent Examiner,
Art Unit 2193
May 16, 2007

[illegible]
MENG-AL T. AN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100